03

DAO con INSERT del producto

Transcripción

[00:00] Hola. En la clase anterior creamos la representación de la tabla producto en la clase de modelo en la aplicación para que podamos realizar la persistencia de los datos. Pero acá en el productoController, nosotros vimos que siempre que vamos a realizar cualquier tipo de operación en la base de datos, nosotros tenemos que abrir una conexión, crear un statement, tenemos un montón de código repetido. ¿Cómo podemos mejorar eso?

[00:29] Nosotros podríamos crear una clase que mantenga todos lo que es necesario, como la conexión de la base de datos para persistir el producto y solamente tengamos que basar el objeto como parámetro. Sería una clase más específica para realizar operaciones en la tabla de productos. Vamos a probar esta idea viendo una clase llamada persistencia producto. ¿Qué les parece?

[00:51] Esta clase vamos acá, a abrir acá y crear un paquete nuevo. Vamos a poner un paquete new package com.alura.jdbc.persistencia y vamos a crear una nueva clase que se llama PersistenciaProducto. Ahora sí, con CamelCase. Acá voy a agrandar un poquito. Esta clase va a tener como atributo la conexión, entonces vamos a tener acá un private Connection, ya importo acá, con; y esta conexión nosotros vamos a asignarla desde un constructor.

[01:40] Entonces vamos a tener acá un PersistenciaProducto, un constructor, que recibe una (Connection con) también. Y nosotros la vamos a asignar a nuestro atributo de connection que tenemos en la clase. ¿Ahora qué vamos a hacer? ¿Qué estábamos diciendo? Que íbamos a registrar un nuevo producto, entonces podemos crear acá un método llamado guardarProducto.

[02:13] Entonces vamos a hacer así, vamos a hacer un public void guardarProducto y nosotros recibimos como parámetro un producto. Perfecto. ¿Qué lógica agregamos a este método? Importo acá el producto primero. Acá en este método, vamos a agregar la lógica de productoController, entonces todos este método que tenemos, vamos a copiar acá y pegar acá en la clase de persistenciaProducto.

[02:47] Lo mismo con el ejecutar registro, porque es el método que estamos finalmente haciendo la persistencia. Entonces esta lógica también viene para acá y lo que tenemos en productoController, podemos borrar esto, ya que lo movimos para persistenciaProducto. En esta lógica de guardar, nosotros podemos eliminar esta información, toda esta lógica y podemos hacer lo siguiente.

[03:16] Vamos a, en lugar de eso, vamos a instanciar un new PersistenciaProducto pero tenemos que enviar una conexión. Entonces vamos a hacer lo siguiente, vamos a hacer un (New ConnectionFactory().recuperaConexion()); vamos a hacer esto y vamos a asignar eso a una variable y este persistenciaProducto vamos a llamar al método persistenciaProducto.guardarProducto(producto); ahí está. Ya dejamos la lógica mucho más sencilla en el controller.

[04:08] Y si nosotros queremos hacer esto con el listado de productos, nosotros podemos hacer algo similar a esta lógica también, pero antes de seguir con eso vamos a entender qué está haciendo esta nueva clase que creamos, la persistenciaProducto. Nosotros tenemos acá nuestra aplicación en Java y la base de datos de MySQL y en el medio de este camino estamos agregando una cajita nueva acá, que es la clase persistenciaProducto.

[04:41] Ahí quedó el nombre chiquitito pero agrando un poquito acá. O sea, tenemos la clase persistencia producto que está en el medio del camino para nosotros. Acá estamos haciendo la aplicación, llama a persistencia producto,

que accede a MySQL para realizar la operación de guardar el producto, un producto nuevo en la base de datos.

[05:09] La idea de esta clase es dar todo el acceso a las operaciones de base de datos para la entidad de producto. Entonces esta clase lo que está haciendo es tratar todas las operaciones en la tabla de producto. Entonces su finalidad es acceder a los datos de nuestro objeto.

[05:28] Si nosotros queremos guardar un nuevo producto, la aplicación va a crear un objeto acá, lo va a enviar para la clase persistenciaProducto, que va a ser la operación de insert en el MySQL en la base de datos. Este tipo de clase y su instancia tiene un nombre específico, esta clase de persistencia producto. Ella tiene un nombre específico, ese nombre es Data Access Object, que también es conocido como DAO.

[05:56] Las clases del tipo DAO, y acá voy a dejar arriba el nombre DAO Data Access Object, estas clases del tipo DAO trabajan con las operaciones de acceso a los datos de una entidad. Entonces nosotros podemos decir que esta clase de persistenciaProducto es un DAO que trabaja con la entidad de producto.

[06:28] Entonces nosotros podemos cambiar su nombre acá, podemos decir que en lugar de persistenciaProducto, ella se llama producto DAO porque es la clase que tiene, que realiza el acceso a los datos de producto. Entonces aquí en Eclipse, nosotros vamos a hacer lo siguiente. La clase persistenciaProducto acá está lanzando un errorcito que es de SQLException, está bien.

[07:06] Pero en la clase persistencia producto nosotros podemos cambiar su nombre acá, podemos cambiar su nombre en la clase persistencia producto, hacer un refactor y vamos a llamarla de ProductoDAO. Ahora sí tenemos esta clase con el nombre más alineado con los estándares en los patrones de diseño y listo.

[07:38] Otra cosa que podemos hacer acá para mejorar es, ya que tenemos la conexión, que la recibimos desde el objeto, desde el controller, nosotros no

necesitamos más estar instanciando acá esta conexión, porque ya la tenemos en la propia clase. Entonces, esto de acá también podemos cambiar y ya lo estamos utilizando directamente acá en el Try with resources.

[08:06] Lo declaré como final acá para que no se rompa y ya estamos también, ya tenemos la conexión del DAO y lo estamos utilizando en el guardar producto. Ya sacamos esta responsabilidad del propio método de estar solicitando una conexión para el data source.

[08:21] Otra cosa que podemos hacer acá es cambiar el nombre del paquete que estamos. Ya no es más persistencia, nosotros podemos decir que el nombre de este paquete es DAO también, ya que vamos a tener todas las clases DAO acá.

[08:38] Entonces voy a guardar acá estos cambios y ahora si va a ser com.alura.jdbc.dao en donde vamos a estar guardando las clases DAO de nuestro proyecto. O sea, esta clase acá de producto DAO va a tener centralizadas todas las operaciones que se refieren al acceso a la tabla de producto. Van a estar ahí dentro de la clase producto DAO.

[09:06] Otra cosa que vamos a estar haciendo acá es, en el productoController estuvimos haciendo el persistenciaProducto, vamos a cambiar su nombre ya que no es más persistenciaProducto y si productoDAO. Hacemos acá un refactor, rename y ponemos productoDAO perfectamente.

[09:27] Y el método guardarProducto, ya que sabemos que este objeto solamente maneja las operaciones sobre la tabla de productoDAO, sobre la entidad de productoDAO, podemos decir que es el guardar. Entonces, productoDAO.guardar y ya sabemos que está haciendo una operación de guardar un producto nuevo en la base de datos.

[09:51] Listo. Más allá de realizar nuestro código una vez más, aprendimos un nuevo patrón de diseño que es el DAO. La finalidad de este patrón de diseño es tener un objeto que tiene como responsabilidad acceder a la base de datos y realizar las operaciones necesarias sobre la entidad.

[10:10] ¿Entonces este patrón sirve solamente para acceder a bases de datos? No necesariamente. Podemos acceder a cualquier fuente de datos con este patrón. La idea es justamente centralizar las operaciones en él para evitar la replicación de código. Agregamos una capa de persistencia a nuestra aplicación y ella va quedando cada momento más ordenada y completa, muy similar a las operaciones del mundo real.

[10:36] En la próxima clase nosotros vamos a finalizar la refactorización del código moviendo el resto de las operaciones del listado borrado y modificación para el DAO.